Chapter 4: Renovation

4. 1. Leveraging Planning Efforts

Planning the Year 2000 conversion effort was discussed in **Chapter 2 Application Analysis**. After the planning phase, renovation can begin. Because Year 2000 is such a pervasive issue that requires an extensive amount of communication between technical team members, it is imperative to have a comprehensive plan in place before proceeding. It is equally as important to adhere to the plan. Unforeseen circumstances will force changes to the plans. Once conversion has begun, any change must be documented and widely communicated to all affected parties. Communication and coordination are the key challenges to the Year 2000 renovation effort.

4. 2. Establish Source Code Baseline

The development team establishes a baseline before Year 2000 renovation begins. The team must put in place a solid process for tracking Year 2000 and normal production changes to that baseline.

By having this baseline, the changes can be determined and undocumented changes can be identified. If parallel development was used, the two revised sets of code can be compared to the baseline, then the change sets can be identified and merged to create the Year 2000 compliant version.

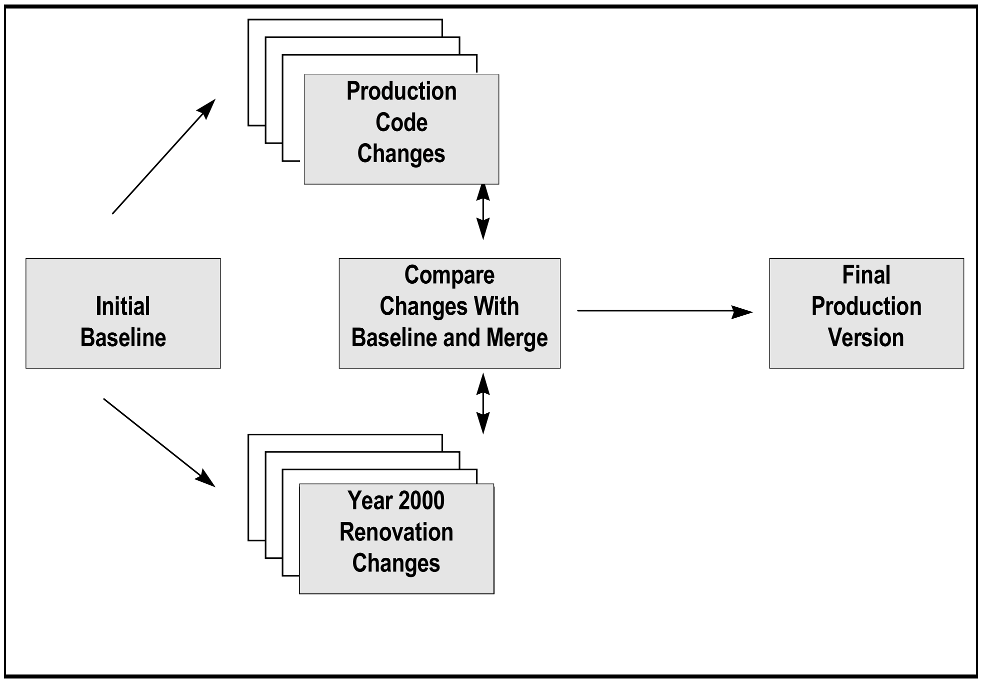
Figure 4-1 depicts an established baseline from which two processes begin; production code changes (top); and Year 2000 renovation changes (bottom).

At the end of the renovation effort, the changed production and renovation code are compared to the baseline and the two versions are merged into a final version.

There are two stages to these merges:

1. The physical merging of the two sets of code. This can be automated with a report showing change collisions.
2. A validation that the logic changes on the two paths do not interfere with each other. This requires testing and validation.

Figure 4-1: Tracking Parallel Development Activities


Production Code Changes
Initial Baseline
Compare Changes With Baseline and Merge
Final Production Version
Year 2000 Renovation Changes

The Year 2000 Project Office is exploring several options for how to best track parallel development or emergency changes to production code. The following information is based on the work to date for this ongoing effort:

➤ For the Unisys systems, application development teams can use Downdater to track changes to code. The Symbolic Stream Generator (@SSG) can be used to merge the changes and produce a list of changes which collide. See the **HUD Year 2000 Tools Overview, Document F** in the **Reference Library** for a description of this Unisys Change Analysis Method.

➤ For the Hitachi platform, HUD has purchased Computer Associates' companion tools ENDEVOR and Parallel Development Manager. See **Document F** of the **Reference Library** for additional information about these tools or send a message to the Team 2000 cc:Mail box at Team_2000@HUD.gov.

➤ On the PC/LAN, tracking parallel development is easier because the applications are smaller and therefore less time is involved. PVCS is a good over-all tool for this task, although each application's case should be considered on an individual basis.

4. 3. Conversion

4. 3. 1. Design Rules

Design rules were established in the high level application analysis phase. They determine the use of either field expansion, windowing, or a combination of the two techniques. In addition to these overall design rules, a conversion strategy must be determined for each date impacted field/variable. See **Chapter 2**, **Application Analysis** for more information.

4. 3. 2. TransCentury Date Routines

TransCentury date routines are the HUD standard for date manipulation on both the Hitachi and Unisys mainframes. The routines are used for obtaining an 8-digit system date. For more information on TransCentury Date Routines, see **HUD Year 2000 Tools Overview**, **Document F** in the **Reference Library**.

4. 3. 3. Programmer's Checklist

A valuable tool during the renovation phase is a programmer's checklist. A checklist:

- ➤ Ensures consistency of approach;
- ➤ Maintains quality deliverables;
- ➤ Establishes, monitors, and maintains a repeatable process; and
- ➤ Assists renovation teams in training junior-level programmers.

Refer to the **Reference Library**, **Document D** for the **Year 2000 Checklists, Worksheets, and Templates**.

4. 3. 4. Input from Impact Analysis Tools

Impact analysis tools produce reports of varying levels of detail. Some reports from analysis tools are useful as input to the renovation phase. For example, the System Vision 2000 impact analysis tool (for Hitachi) provides reports listing every impacted line of code. This report can be given to a programmer as a reference when renovating code. System Vision 2000 also has an Interactive System Productivity Facility (ISPF) utility that walks the user through the source code online to support the change process. For more information on Impact Analysis Tools, refer to the **HUD Year 2000 Tools Overview**, **Document F** in the **Reference Library**.

4. 3. 5. Documentation

There are multiple levels of documentation required during the renovation phase. Below are some major categories:

- ➤ Status,
- ➤ Program improvement/changes,
- ➤ User impact/change specification, and
- ➤ File and database structure changes.

4. 3. 5. 1. Status Documentation

It is important that status information is shared with points of contact for interfacing systems, especially if renovation efforts are dependent on these systems. The STATUS 2000 database serves as the central point for tracking progress and reporting status on all HUD Systems (See **STATUS 2000 Database Information**, **Document I** in the **Reference Library** for user information).

4. 3. 5. 2. Program Improvement/Changes

This documentation must contain a change number and may be recorded on the sample Program Improvements/Changes form available in the **Reference Library**, **Document D**. This form is designed to collect information such as:

- Program ID,
- Stage of Testing, and
- Improvement/Change.

All changes must be identified in the code. In order to comply with the known future audit, the programmer's change log in the remarks section at the beginning of the program/module must clearly document all changes made, identify them as Year 2000 changes and identify the programmer's initials, the date the change was made, and the change number. Programmers must also insert comment/change lines before and after each change in application code.

The National Bureau of Standards has studied the issue of documentation. See Federal Information Processing Standards Publication 38 (FIPS PUB 38), *Guidelines for Documentation of Computer Programs and Automated Data Systems*.

4. 3. 5. 3. User Impact/Change Specification

The standard user change approval process must be followed for any change in functional performance and/or look and feel of an application due to Year 2000 renovation.

4. 3. 5. 4. File and Database Structure Changes

Any file and/or database table changes must be reflected in technical documentation. Database impacts must be communicated to the responsible database administrator (DBA) and approved.

4. 3. 5. 5. Documentation of Changes

Software changes should be documented using standard techniques as described in HUD's Systems Development Methodology (SDM).

Given the likelihood of audits, all changes must be clearly recorded. At a minimum, the following should be placed in any altered program:

- A description of the changes, the date and the name of the person making the change must be added to the REMARKS section of a COBOL program.
- Every changed line of code must have initials and date placed in columns 73-80.

4. 4. Database Redesign and Reorganization

The key objective of this task is to identify all changes to the databases of a release and complete those changes. The primary change is to expand year fields from two to four digits, and to populate the extra two digits with the appropriate value for the century. However, additional changes might be required. For example, the definition of the sort field or index field might be modified. In extreme cases, the database physical layout might be changed if record length or database size constraints are exceeded or if performance is degraded beyond an acceptable level.

Below are the general steps involved in making Year 2000 design changes to logical and physical database designs:

1. Revise design for database structures,
2. Review and refine database design, and
3. Verify, rebuild and convert data.

4. 4. 1. Revise Design for Database Structures

For the logical database redesign, the functional analyst must identify all date columns from application tables that must be expanded, as explained in **Section 2.4.1**, **Identify Date Field Changes**. If a column to be expanded or changed is a key field, existing relationships between tables must be verified after the column expansion. Changes to logical design are performed by the functional analyst.

The Database Administrator (DBA) works with the functional analysts to coordinate changes. The DBA evaluates any required redesign of the physical database. The current grouping of tables into physical storage must also be evaluated. The DBA must determine if reallocations of free space and buffer sizes are needed.

For expanding columns, the data element descriptions and table space descriptions must be modified to reflect Year 2000 changes. Create a list of all the program components that are affected by table expansion. For every data file or database, the analyst specifies:

- If the data file or database is to be expanded in this release,
- The data conversion template to be used for this expansion,
- If the interface is to be bridged in this partition,

- All programs that require bridging and whether the program reads and/or writes from/to this interface, and
- The bridge template to be used.

The DBA is responsible for creating the new copybook as a result of the field expansion.

4. 4. 2. Review and Refine Database Design

For cases where changes are other than field expansion (for example, collapse date-related columns) the DBA executes and tests changes in a non-production environment. A traffic analysis must be performed. Note that if change is only to physical design, traffic analysis testing may be delayed until system testing. Database options (for example, space allocations, table space partitioning) must be reviewed and possibly updated. The redesign process must be repeated until an acceptable performance level is reached. Next, the design is reviewed with the corporate database administrator.

4. 4. 3. Verify Validity of Incoming Data

Systems routinely edit incoming data files to verify that they meet certain criteria. These edits provide assurance that the data is reasonable and that it conforms with parameters meant to ensure error free processing.

With so many alternatives available to programmers in deriving four-digit year representations, it is quite possible that the source system(s) may be sending incorrect dates or that it may have incorrectly calculated a data field as a result of incorrectly derived dates (e.g., age figures are incorrect because calculations are being performed with incorrectly derived century digits). This issue grows larger if the source is from an external trading partner where the IT culture is less predictable and less likely to conform with internal assumptions and conventions.

It is especially important, therefore, that the developer of the receiving system re-examine the error processing procedures and possibly revise them to obtain corrections if corrupt data is detected, or if no file is received. This will likely necessitate contact with the data sender(s). Also, users may need to be consulted to completely assess the business impact of data corruption or processing disruptions brought about by corrupt data.

4. 5. Data Conversion and Bridge Program Templates

A Year 2000 implementation may involve hundreds of data files and databases. If no tool exists to perform these transformations, the implementation of hundreds of data conversion programs and bridge programs becomes necessary. Although this may be a very large effort, the designs of these programs are usually quite simple and similar to each other.

4. 5. 1. Three Programs for Designing Templates

The objective of this task is to design a set of easy-to-use templates to assist in the development of data conversion and bridge programs. A Year 2000 implementation may require up to three different types of programs:

1. A **data conversion program** is needed when there is a design change to a permanent data file or database, most typically the expansion of one or more year fields from two to four digits. Other changes might involve sorting and indexing.
2. A **bridge program** lets the sending and receiving functions modify their code on separate schedules. A bridge is needed when there is a design change to a data file or database, whether or not it is permanent, and the programs which interface with this file are being converted in different releases. HUD's approach is to build external bridges not coded within a program and run as a separate step. This lets programs effectively process exchanged data.
3. A **data extraction program** can be used to create test data from existing data. This case can be applied whether or not the file is permanent, as long as it is an input to or an output from a test. The program may perform year field expansions, and also generate modified dates, such as 21st century dates.

4. 5. 2. Objectives and Structure of Templates

The objective of using templates is to encapsulate the common design elements and to make it as simple as possible to develop each individual data conversion or bridge program.

Each template includes the following:

- A file containing a complete, compilable, executable program, which does the data conversion of a single file or implements a single bridge. A sample of the conversion program is provided in the Reference Library. The program contains highly visible comments describing the parts of the code which must be replaced, and additional comments providing instructions on how to customize the code for a specific case.
- A complementary document which describes:
  - When to use the template (for example, which types of files it can be used on), and
  - How to use it.
- Complementary templates of Job Control Language (JCL) or other files (for example, the JCL to actually execute the data conversion program). If the amount of JCL that can be re-used is truly minimal, then include the documentation on the JCL requirements.

This task consists of the following subtasks:

- ➤ Identify the implementation technology (or technologies) to use. For example, File-AID, Data Xpert, COBOL, etc.;
- ➤ Determine the set of templates needed for VSAM, DB2 and IMS DB;
- ➤ Write design specifications for each template.

4. 5. 3. Automated Tools and Templates

If an automated tool is available (such as File-AID), then the tool might be used instead of a program. The program template would be replaced by a template for tool inputs, and the JCL template would be replaced by a JCL template which executes the tool. A good tool might reduce the cost of developing data conversion and bridge programs significantly.

4. 5. 4. Multiple Templates

A Year 2000 implementation project may need multiple data conversion and bridge templates, with different templates designed for different types of data files and databases. This could include basic sequential files or more complex files (such as sorted vs. unsorted, indexed vs. not indexed, single vs. multiple record types).

The number of templates required may also depend on the interface design rules selected for this project. For example, if the design rule is to minimize changes to files, then there may be no requirement for bridge programs, and therefore no bridge program templates. Conversely, if the design rule is to always expand year fields from two to four digits, then a comprehensive set of templates might be required.

4. 5. 5. Template Issues for Consideration

Some issues to consider when designing the templates include:

- ➤ Would conversion requirements be satisfied by creating SYNCSORT or CLIST routines?
- ➤ Is there any utility provided by the application's DBMS or File Management product that can be used instead?
- ➤ Must the conversion program update the table directly or only update a file that could be uploaded as a table?
- ➤ Must the conversion program convert specific records or convert the whole table in one run?
- ➤ Most of these programs are used only a few times, so performance and ease of use are less critical than for the application itself. However, reliability and integrity controls are just as important.
- ➤ The conversion or bridge program templates and their I/O modules must be as generic as possible (for example, one conversion program that is able to convert any data file with varying lengths).
- ➤ Design templates must accept different parameters.

4. 5. 6. Conversion/Bridge Program Specification Documents

The following are the documents that may be included in conversion or bridge program specification. The exact structure of the specifications depends on the organization of the team, the language, and the tools used:

- Module abstract,
- Module flow diagram,
- Conversion requirement/flow,
- Structure chart,
- Design issues document,
- Potential test conditions,
- Input file/table layouts, and
- Output file/table layouts.

See the **Sample Conversion/Bridge Program Template** in the **Reference Library, Document D**.

4. 5. 7. Build Conversion and Bridge Templates

Each renovation team is responsible for developing a set of data conversion and bridge program templates. There needs to be a template for each type of data set, (such as sequential or VSAM, or data base). When possible, the template uses File-AID (on the Hitachi) as the actual data converter; otherwise a COBOL shell program with generic logic for reading input files and writing output files of varying lengths must be provided. Customizable JCL will be included as part of the template.

4. 6. Design Data Conversion and Bridge Programs

Data conversion and bridge programs are used to convert data from one record layout to another. These programs are submitted to run once during file and/or data base conversion.

4. 6. 1. Executing Bridge Programs

Bridge programs are executed during program flow. It becomes part of the normal operating environment, at least for a period of time (for example, programs created to bridge between a new program and a program in-between release are deleted from batch flow once both programs are using new release. Note that programs created to bridge between a new release and other systems –especially those external to HUD, such as systems of business partners– may become a permanent part of batch flow.)

4. 6. 2. Executing Conversion Programs

The conversion occurs each time data is passed between old (unchanged) programs and new (changed) programs or between a program and external data using a different format.

In most cases, Year 2000 data file and database conversions require logic to populate century values that would be linked with the old date value. Depending on how easy it is to identify the century for a certain year and what is being converted, the data conversion may be a query (for example, SQL or QMF), may be a routine or a series of commands (such as CLIST, SYNCSORT, File-AID/MVS, File-AID/XPERT), may be one separate module (such as a COBOL program) or a combination of these.

4. 6. 3. Steps and Factors of Conversion/Bridge Programs

Below are the general steps and factors involved in designing conversion and/or bridge programs, usually carried out by the functional analyst:

- **Review existing documents**. Study the updated data file and database design documents and revised copybooks, then design a conversion program to convert one file or table at a time.
- **Select a data conversion or bridge program template**. Choose the template which most closely matches the conversion or bridge required (for example, one determined by the type of file or database and the types of records to be converted).
- **Write a data conversion/bridge program specification, identifying the customizations needed for the template**. In many cases, the specification is completely encapsulated by the differences between the old and new copybooks, together with a specification of the century values to be put in the expanded year fields.
- **Include data verification routines in data conversion and bridge programs**. Data conversion programs must have standard error reporting functions flagging invalid dates. Extensive knowledge of the data is crucial for this task. Database administrators or users who regularly work with the data are good resources to utilize here.

4. 7. Build Data Conversion and Bridge Programs

This task entails programming and unit testing of conversion and bridge programs and their integration with the updated source code modules. The testing techniques for these programs are not really different from the mainstream application programs. They are called out here because Year 2000 implementation projects usually have them and they have an impact on the workplan and costs of the project. Furthermore, Year 2000 conversion requires a substantial amount of effort because of the sheer volume of files and programs affected by the century date change.

Note that the detailed Year 2000 test conditions in **Chapter 5**, **Testing**, do not apply. Moreover, there are no separate tests for old (without Year 2000 logic) and new (with Year 2000 changes) versions of the programs since there is only one version of data conversion and bridge programs to test.

4. 7. 1. Basic Logic for Year 2000 File Changes

The basic logic for Year 2000 file changes may include the following:

- Expansion of the year or date field to accommodate the 2-byte century.
- Defaulting of hard-coded 19 to the century field of the date or year if the system processes transactions within the 20th century only.
- Formatting of the century of date or year fields with 21st century values to test if the program can handle dates beyond 1999.

4. 7. 2. Basic Logic for Bridge Programs

Year 2000 bridge programs are also used to convert old and new date impacted file formats in different releases. They can have any of the following logic:

- Reading an input file with a 6-byte date field (YYMMDD) and writing to an output file with 8-byte date field (CCYYMMDD), or vice-versa, going from 8-byte to 6-byte.
- Moving a 2-byte year field (YY) to a 4-byte year field (CCYY) for programs that process the new file format.
- Moving a 4-byte year field (CCYY) to a 2-byte year field (YY) for releases requiring the old file format.

4. 7. 3. Completing Data Conversion/Bridge Program Task

The following are the steps involved in completing this task:

- Review the **Data Conversion/Bridge Program Specification** (see **Section 4.5.6** for background).
- Code the program based on the template named in the specification.
- Check the source code for syntax errors. Edit, if necessary, and recheck until ready for unit test.
- Test bridge programs with the updated source modules during the application's system test phase.

4. 8. Execute Data Conversion and Implement

4. 8. 1. Create Data Conversion Plan

This plan includes all steps necessary for conversion of data file and database entities. The plan includes an inventory of all reformatted files and database entities and a schedule. The schedule must be communicated with and approved by the responsible DBA and those in charge of all affected systems. The data conversion plan must include a test plan for data conversion programs/routines, as well as unit testing and a mock conversion exercise.

4. 8. 2. Create Inventory of Reformatted Files

When creating the inventory of reformatted files, pay special attention to:

- Static files,
- Files that receive I/O,
- Cycled files, and
- Data capture files.

4. 8. 3. Schedule Data File/Database Conversion

A critical element in the implementation plan is the scheduling of the conversion of data files and databases to be changed. This is especially important for databases accessed by interactive applications.

The sequence of steps are as follows:

- Shut down all systems which access the database to be converted.
- Convert the database from the old to the new format. For a large database, this step could be quite lengthy.
- Install the new versions of the systems which accept the new database format.
- Restart the systems.

4. 8. 4. Conversion Planning and Year 2000

The conversion planning phase in a Year 2000 engagement is broadly similar to that of a typical systems development project. Hence, normal conversion considerations apply.

4. 8. 4. 1. Year 2000 Conversion and Implementation Issues

What makes conversion and implementation of systems for Year 2000 unique from the traditional systems is the criticality of a well-timed implementation of the conversion and implementation schedule. Since the implementation would involve a system for a Year 2000 program, it is apparent that the system has to be installed sometime before the 21st century, depending on the Year 2000 failure dates.

4. 8. 4. 2. Steps for Year 2000 Installation Planning

The following steps must be undertaken for efficient and effective installation planning:

- ➤ Refer to the overall Year 2000 Renovation Schedule to create a realistic conversion and implementation schedule and estimates.
  - It can be in a workplan format that defines required tasks and the associated effort, time frames, and area of responsibility.
  - It may also include implementation assumptions and dependencies specific to timing, scope, and so on. This is prepared by the implementation team with the technical group.
- ➤ A plan is developed for each application system or grouping of application systems to be rolled-out.
- ➤ Provide enough leeway or buffer in the Contingency Plan for the required time, effort and staffing during system installation.
  - Contingency plans are created so that target dates are not missed in case of an equipment breakdown, major personnel problems, or any other unforeseen occurrences.
  - A contingency plan includes fallback procedures, manual handling of volume data, equipment backup arrangements, and the continued use of the old system.

(This page has been left blank intentionally.)